The chess game

Documentatie limbaj natural

1. Descrierea si logica proiectului

Descrierea generala a aplicatiei

Scopul proiectului per total este sa se creeze un sistem capabil sa evolueze situatiile din timpul unui joc de sah si sa dea sugestii pentru miscari optime la orice punct al jocului, indicand posibile greseli, generand comentarii in limbaj natural asupra jocului. Cu alte cuvinte, proiectul face o aplicatie capabila sa analizeze un joc de sah si sa propuna cea mai buna mutare de la un anumit punct al jocului pentru a ajuta pe cineva sa invete sa joace sah sau sa isi imbunatateasca abilitatile.

Sistemul foloseste anumite strategii cunoscute pentru a alege cea mai buna miscare de la un anumit punct si pentru a determina cand au avut loc anumite greseli.

Asadar, aplicatia va avea doua functionalitati, analiza mai multor mutari de sah consecutive(poate fi si un joc intreg) si sugestia miscarii optime pentru o anumita stare a jocului.

Descrierea modulului de limbaj natural

Modulul de limbaj natural consta in generarea de comentarii pentru analizarea mutarilor jocului si generarea de comentariu pentru a descrie cea mai buna mutare la un anumit punct al jocului.

In primul caz, adica analiza mai multor mutari, pentru a putea face asta modulul de limbaj natural primeste de la modulul de inteligenta artificiala o stare initiala a tablei de sah si o lista de mutari, fiecare mutare avand un scor, calculat de modulul de inteligenta artificiala in functie de cat de buna a fost mutarea. De asemenea, pe langa scoruri, la unele mutari modulul de inteligenta artificiala va mai calcula si posibilie varinate de mutari dupa unii algoritmi implementati de ei. Pe baza scorurilor noi vom decide cat de buna a fost mutarea si, de asemenea, scorurile se vor folosi si in cazul in care mutarea are variante pentru a calcula daca vreo varianta la mutarea aceasta este mai buna decat mutarea reala facuta, si pentru a determina mai exact care este cea mai buna varianta. Daca o varianta este mai buna cu un anumit procentaj de scor decat mutarea reala facuta, se va genera un comentariu in care se va spune ca a fost facuta o greseala si se va descrie ce s-ar fi intamplat daca s-ar fi facut mutarea in functie de cea mai buna varianta. De asemena, la fiecare mutare relevanta jocului(capturarea de piesa, promovare pion, sah etc.) se va genera un comentariu in care se va zice exact ce s-a intamplat. Pentru a face acest lucru modulul de limbaj natural va parcurge jocul si va determina ce s-a intamplat pe parcursul jocului cu ajutor unor librarii specifice.

In al doilea caz, alegerea mutarii optime pentru o anumita stare a jocului, vom primi de la modulul de inteligenta artificiala o anumita stare a tablei de sah si posibile variante de mutari pentru acea stare a jocului(adica o lista de mutari de la acel punct facute de ambele parti, nu doar o mutare). Aceste mutari vor avea si ele un scor si pe baza scorului noi vom calcula care este cea mai buna mutare la acel punct(in functie de mutarile ce vor avea loc in viitor, nu doar in functie de prima mutare) si vom descrie ce se intampla de-a lungul acestei mutari pentru a se intelege de ce este cea mai buna mutare.

De asemenea, am primit de la modulul de inteligenta artificiala detaliile despre ce inseamna scorul, adica ce inseamena o mutare buna,foarte buna, proasta, mutarea care ii la un pas de sah mat etc.

Notatii ale jocului de sah folosite

Pentru a reprezenta o stare a tablei de sah folosim notatia Forsyth–Edwards Notation (**FEN**). Aceasta notatie ofera toate informatiile necesare pentru a reporni un joc de sah de la o anumita pozitie.

Un FEN defineste o pozitie particulara a unui joc de sah, folsoind un sir ASCII de caractere.

Un FEN contine sase parti, separate fiecare printr-un spatiu. Aceste parti sunt:

1. Pozitionare pieselor(din perspectiva albului).
2. Culoarea care muta - w pentru alb si b pentru negru.
3. Posibilitatea de a face rocada. Daca nici o parte nu poate sa faca rocada este - , altfel, are una sau mai mult litere: K (albul poate sa faca rocada in partea regelui), Q(albul poate sa faca rocada in partea reginei), k(negrul poate sa faca rocada in parte regelui) si/sau q(negrul poate sa faca rocada in parte reginei).
4. Casuta in care se poate face en pasant sau – daca nu se poate face.
5. Halfmove clock: acesta este numarul de mutari de cand a fost ultima capturare sau avansare a pionului. Este folosita pentru a determina daca se ajunge la egal sub regula de 50 de mutari.
6. Numarul total de mutari pana la acel pas

Exemplu de pozitie initiala pentru tabla de sah:

rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

Exemplu de stare pentru tabla de sah:

rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 0 1

Pentru a reprezenta o mutare a unei piese este folosita o variatie a notatiei Portable Game Notation (**PGN**). Aceasta notatie este reprezentata din 5 sau 6 caractere. Primul caracter reprezinta culoarea care face mutarea(w pentru alb si b pentru negru), al doilea impreuna cu al treilea reprezinta pozitia pe care este

piesa pe tabla(ex. a1), al patrulea impreuna cu al cincilea reprezinta pozitia pe care este mutata piesa(ex. a2) iar al saselea este prezent doar in cazurile in care pionul este promovat in alta piesa si reprezinta piesa in care este promovat acel pion(ex. q).

Ex. mutare simpla: wg2g3

Ex. mtuare cu promovare: wa7a8q

Tip proiect, limbaj folosit

Proiectul pe partea modului de limbaj natural este realizat in limbajul de programare Java 8 sub framework Spring si este un proiect de tip Maven.

7. Detalii implementare proiect

Structura generala modul limbaj natural

Proiectul este o aplicatie de tip web si consta in doua requesturi web de tip POST, primul pentru cazul in care se propune cea mai buna mutare de la un anumit punct al jocului, la endpoint-ul **/optimal-moves** , si al doilea pentru cazul in care se primeste o lista de mutari si se ofera analiza acestor mutari prin intermediul unor comentarii pentru mutarile cu un impact asupra jocului, la endpoint-ul **/game**.

In body-ul acestor post request-uri primim inputul descris mai sus de la modelul de inteligenta artificiala dupa care il vom mapa la structurile pe care le vom folosi noi.

Urmatorul pas consta in decorarea mutarilor cu metadate necesare pentru generarea comentariilor. Pentru a face asta se parcurge jocul de sah cu ajutorul

unei libarарii de Java specifica jocului de sah si la mutarile cu un impact asupra jocului se adauga metadata necesara pentru ceea ce s-a intamplat in mutarea respectiva (ex. daca se ia o piesa se adauga la lista de metadate a mutarii metadata PieceTakenMetadata care va indica faptul ca la acea mutare a fost luata o piesa si, de asemenea, va contine si numele piesei luate). Decorarea cu metadate se face atat pentru lista de mutari principale(adica mutarile care au avut loc in joc) cat si pentru posibilele variante de mutari mai bune la o anumita mutare si pentru cazul alegerii mutarii optime intr-o anumita pozitie a tablei de sah.

Dupa asta, in cazul alegerii mutarii optime la un anumit punct al jocului de sah, se va calcula care este cea mai buna mutare data de algoritmi utilizati de cei de la modulul de inteligenta artificiala in functie de scorul mutarilor si se vor adauga comentarii pentru variantele de mutari care vor avea scor maximal in care se va descrie ce se intampla pe parcursul acelei variante de joc, deci va descrie de ce acea mutare este cea mai buna mutare posibila din lista de mutari primite.

In cazul analizei unei liste de mutari, se vor adauga comentarii pentru fiecare mutare care contine una sau mai multe metadate necesare pentru adaugarea de comentarii(ex. daca o mutare are PieceTakenMetadata se va genera un comentariu in care se va zice ca aici a fost luata piesa x si se va adauga la comentariul mutarii). De asemnea, pe langa comentariile standard cu ce se intampla pe parcursul mutarilor unde ne va fi oferit de la modulul de inteligenta artificiala variante de mutari posibile mai bune dupa algoritmi folositi de ei, noi vom calcula daca vreo mutare este mai buna decat mutarea care a avut loc(in functie de scorul oferit de ei, de asemena tinem cont de mai multe mutari care au loc in viitor) iar daca se intampla acest lucru vom genera un comentariu in care se va spune ca a avut loc o greseala si pentru a argumenta greseala vom descrie ce s-ar fi putut intampla daca am fi facut alta mutare(una aleasa de algoritmi folositi de modulul de ai). Pentru descrierea mutarii se foloseste aceeasi procedura ca pentru generarea comentariului pentru alegerea mutarii optime.

Dupa pasurile respective pentru ambele cazuri vom parcurge listele de mutari si le vom mapa la outputul pe care va trebui sa-l oferim modulului de interfata grafica si vom returna aceste structuri ca rezultate la requesturilot POST respective.

Exemple de input primit de la modulul de inteligenta artificiala

Input cazul mutari optime


```
{
  "initialStateFEN": "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1",
  "variants": [
                {
                   "moves": [
                      {
                         "move": "wd2d4",
                         "score": 0.01
                      },
                      {
                         "move": "bg8f6",
                         "score": 0.0
                      }
                   ],
                   "algorithmName": "MinMax",
                   "strategyNames": [
```

```
                "PieceRemained",
                "AttackPieces"
            ]
        },
        {
            "moves": [
                {
                    "move": "wc2c4",
                    "score": 0.0
                },
                {
                    "move": "bg8f6",
                    "score": 0.01
                }
            ],
            "algorithmName": "AlfaBeta",
            "strategyNames": [
                "PieceRemained"
            ]
        }
    ]
}
```

Input caz analiza lista de mutari


```
{
   "initialStateFEN": "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq
- 0 1",
   "mainVariant": {
      "moves": [
         {
            "move": "we2e4",
            "score": 0.01,
            "variants": [
               {
                  "moves": [
                     {
                        "move": "wd2d4",
                        "score": 0.01
                     },
                     {
                        "move": "bg8f6",
                        "score": 0.0
                     }
                  ],
                  "algorithmName": "MinMax",
```

```
      "strategyNames": [
        "PieceRemained",
        "AttackPieces"
      ]
    },
    {
      "moves": [
        {
          "move": "wc2c4",
          "score": 0.0
        },
        {
          "move": "bg8f6",
          "score": 0.01
        }
      ],
      "algorithmName": "AlfaBeta",
      "strategyNames": [
        "PieceRemained"
      ]
    }
  ]
```

```
},
{
   "move": "be7e5",
   "score": 0.01,
   "variants": [
      {
         "moves": [
            {
               "move": "bf7f5",
               "score": 0.01
            }
         ],
         "algorithmName": "AlfaBeta",
         "strategyNames": [
            "PieceRemained",
            "AttackPieces"
         ]
      },
      {
         "moves": [
            {
               "move": "bc7c6",
```

```
                    "score": 0.01
                },
                {
                    "move": "wd2d4",
                    "score": 0.01
                }
            ],
            "algorithmName": "MinMax",
            "strategyNames": [
                "PieceRemained"
            ]
        }
    ]
  }
}
```


Mapare

Pasul de mapare este unul destul de simplu si consta in inputul primit de la modulul de inteligenta artificiala si punerea acestuia in niste structuri de date in care se vor putea adauga comentarii si metadate(lucruri necesare pentru modulul nostru).

De asemenea, dupa ce generam comentariile necesare, trebuie sa mapam structurile noastre de date la outputul necesar pentru modulul de interfata grafica, adica niste structuiri care contin in mare doar comentariile mutarilor(fara scoruri, metadate etc).

Metadatele cu care este decorata fiecare mutare de sah

- **pieceTaken** este un camp String in modelul care va fi null daca nici o piesa nu a fost luata la aceasta mutare si o sa aiba numele complet al piesei care a fost luata in caz contrar
- **castlingState** este un camp String care contine kq in caz ca se poate face rocada in ambele parti(partea dinspre rege si regina),k in caz ca se poate doar in parte dinspre rege, q in caz ca se poate doar in partea dinspre regina si null in caz ca nu se poate face deloc rocada
- **moveGrade** este un camp de tip integer care reprezinta cat de buna este mutarea - 0 pentru stare de de echilibru, 1 pentru o miscare buna, 2 pentru o miscare foarte buna(pentru miscarile rele opusul -1,-2)
- **chessPieceName** este un camp de tip String care reprezinta numele complet al piesei mutate la aceasta mutare ex: King, Queen (de asemenea eu ma gandeam ca ar fi bine ca numele sa fie scrise mereu cu litera mare)
- **color** este un camp de tip String care reprezinta culoarea piesei mutate (White sau Black)
- **enPassant** este un camp de tip boolean care va indicat daca dupa mutarea facuta adversarul va poate face enPassant
- **gameState** este un camp de tip String care va avea valoarea equal daca dupa aceasta mutare jocul a ajuns sa fie egal, checkmate daca aceasta mutarea cauzeaza castigarea jocului si null altfel
- **check** este o lista de stringuri in care se vor gasi piesele care dau sah la rege
- **promotion** este un camp String care o sa contina numele piesei in care promoveaza pionul si null in caz ca nu a avut loc nici o promovare
- **equalScope** este un camp te tip boolean care va fi true daca culoarea actuala joaca in scopul de a face egal(in caz ca a ajuns intr-un dezavantaj

prea mare pentru a mai putea castiga) si false in caz contrar. Decorarea se face doar pentru varianta principala, nu si pentru celelalte variante.

- **preCheckeMate** este un camp de tip boolean care indica daca se poate da sahMat exact dupa mutarea asta(de oponent)

Decorare mutare cu comentariu

Pentru fiecare comentariu o sa se parcurga metadatele si in functie de ce metadate apar o sa se genereze comentarii pentru mutare.

Ex. Daca apare promotion o sa se genereze un comentariu in care sa se zica ca pionul a fost promovat in piesa x la acea mutare

Decorare varianta cu comentarii

Decorarea variantei cu comentarii este folosita atat pentru cazul cand alegi mutarea optima cat si pentru cazul in care se face o greseala la o anumita mutare.

Aceasta decorarea este similara cu cea pentru mutare doar ca genereaza comentarii pentru toata lista de mutari in favoarea culorii care face prima culoare.

Ex. Daca albul muta prima oara in varianta atunci un comentariu ar putea sa arate cam asa: De-a lungul variantei dupa algoritmul x albul ia piesele x,y,z, pierde piesele y si promoveaza un pion in regina.

Exemple de output pentru modulul de interfata grafica

Exemplu output pentru varianta alegerii mutarilor optime


```
[
  {
    "movement": "bh1h8",
```

```
      "comments": [
        "Algorithm used for this moves is NegaMax with strategies [PAWN_ADVANCE_STRATEGY]. ",
        "Black took the following pieces: pawn, queen.",
        "White was checked by Black 2 times."
      ]
    }
]
```


Exemplu output pentru varianta cu analiza unei liste de mutari


```
[
  {
    "moveIndex": 9,
    "evaluation": "!?",
    "comments": [
      "white has captured pawn",
      "The move caused no apparent impact on the game "
    ]
  },
  {
    "moveIndex": 15,
    "evaluation": "!",
    "comments": [
```

```
        "white has captured bishop",
        "The move gave slightly advantage for white side."
    ]
},
{
    "moveIndex": 19,
    "evaluation": "!?",
    "comments": [
        "white has captured pawn",
        "The move caused no apparent impact on the game "
    ]
},
{
    "moveIndex": 34,
    "evaluation": "!?",
    "comments": [
        "black has captured bishop",
        "The move gave slightly advantage for black side."
    ]
},
{
    "moveIndex": 35,
```

```
        "evaluation": "!?",
        "comments": [
            "white side checked the opponent from the piece(s): queen; ",
            "At this move was made a big mistake. With the move wd1a4 this could have happened in the next 3 moves. Algorithm used for this moves is MinMax with strategies [PAWN_ADVANCE_STRATEGY]. White took the following pieces: pawn.Black was checked by White 1 times.",
            "The move gave slightly advantage for white side."
        ]
    }
]
```